Auddard & Co.

Arithmetic Expression Evaluator in C++
Software Requirements Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
<03/10/23>	<1.1>	Discussed and filled out the requirements for our arithmetic calculator in terms of priority for each requirement.	Audrey Pan, Sriya Annem, Morgan Nguyen, Lauren D'Souza, Ella Nguyen, Tanu Sakaray

Table of Contents

Software Requirements Specifications

1. Introduction

The Arithmetic Calculator Software Requirements Specification (SRS) outlines the essential details and specifications for the arithmetic calculator application that we will be creating during this project. This document provides a comprehensive view of the product's perspective, system interfaces, user characteristics, constraints, and dependencies. It also talks about the use cases that we might encounter and different paths for these use cases. Additionally, it presents an overview of the specific functional requirements that this calculator software must fulfill.

1.1 Purpose

The purpose of this SRS is to define the requirements for the development of an arithmetic calculator software application. This document serves as a foundational guide for the designers, developers, and testers involved in creating the software. It outlines both functional and non-functional requirements, ensuring that the software meets the expectations of users and functions smoothly across various devices.

The following people contributed and plan to use this *Software Development Plan*:

- **Project Manager** Audrey Pan uses the software development plan to plan the project schedule and resource needs, and to track progress against the schedule.
- **Project Team Members** Tanushri Sakaray, Sriya Annem, Ella Nguyen, Lauren D'Souza, Morgan Nguyen use it to understand what they need to do, when they need to do it, and what other activities they are dependent upon.

1.2 Scope

The scope of this SRS encompasses the entire lifecycle of the arithmetic calculator software. It provides a high-level view of the product's perspective, including its intended users, interfaces, and constraints. Furthermore, it delves into specific details such as functionality, user input validation, and compatibility requirements. The SRS is a crucial reference point for all stakeholders, from developers to testers, to ensure that the software meets the specified criteria.

Identifying Use Cases:

- identifying the various interactions that users can have with this calculator. This includes the basic operations of the calculator and more advanced ones as well. For each use case we can write descriptions of the steps users can take to perform the arithmetic operations. Specifying input and output. We can also create a diagram with a use case diagram that identifies the actors and their relations to use cases. The actors in this case are the users of the calculator. As we develop this calculator we can identify additional use cases or refine existing ones. It can be an iterative process that evolves as the project does.

1.3 Definitions, Acronyms, and Abbreviations

1. SRS: Software Requirements Specification
2. C++: A programming language used for the development of the software.
3. PEMDAS: An acronym representing the order of operations (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) in mathematics.
4. GUI: Graphical User Interface
5. API: Application Programming Interface
6. Unary Operator: An operator acting on a single operand, e.g., unary minus (-).
7. Tokenize: The process of splitting an input into individual elements for processing.
8. Stack: A data structure following Last-In-First-Out (LIFO) for managing function calls and expressions.
9. Tree: A hierarchical structure used for expression or hierarchical representation.
10. Operator Precedence: The priority order for evaluating operators.
11. Numeric Constant: Fixed numerical value in expressions (integers, potentially floating-point).
12. CLI (Command-Line Interface): Text-based user interaction with software.
13. Error Handling: Managing unexpected situations, e.g., division by zero or invalid input.
14. SDP Version: Document version number indicating revisions.

15. Gantt Chart: Visual project schedule representation.
16. Change Requests: Formal requests for project requirement, scope, or deliverable changes.
17. Configuration Management: Controlling changes to project artifacts, versioning, and naming.
18. UPEDU Process: Unified Process for EDUcation, used for educational purposes.
19. SN (Serial Number): A serial number is a unique code used to identify and track individual items or products.
20. SDP (Software Development Plan): A document outlining the software development project's approach, strategies, and key aspects.

1.4 References

SN	Title	Published Date	Publishing Organization
1.	EECS348: Term Project in C++ Specification	09/Sep/23	Professor Hossein Saiedian
2.	Handling divide by zero error	23/Jan/19	ManasiKirloskar
3.	Microsoft How to make a calculator on C++	3/19/2023	8 contributors
4.	PEMDAS- order of mathematical operations in C++	1/April.2022	Gearbox Labs
5.	Inspiration for GUI for our calculator	22/September/2021	Filipe Salgueiro

1.5 Overview

The arithmetic calculator software is a versatile standalone application designed to run on desktop and mobile devices. It offers basic arithmetic calculations, including addition, subtraction, multiplication, and division. The software includes a graphical user interface (GUI) with buttons for digits, operators, and essential functions like clear, equals, and backspace. Users can input mathematical expressions through button clicks or keyboard input.

The calculator relies on standard hardware interfaces, such as display screens and keyboards. It may also utilize software interfaces, libraries, or APIs for mathematical operations and user interface components. This software does not require external communication interfaces, as it operates locally on the user's device. It should be optimized to run efficiently within the memory constraints of the target devices and provide real-time results.

The primary function of the calculator is to perform arithmetic calculations accurately and efficiently while adhering to mathematical rules and constraints, such as the order of operations (PEMDAS).

While this overview provides a high-level understanding of the arithmetic calculator software, the specific functional and non-functional requirements that guide its development are detailed in Section 3 of this SRS. These requirements cover aspects such as mathematical operations, user input validation, backup and recovery, and compatibility. Furthermore, the document classifies these requirements into essential, desirable, and optional categories, ensuring that the core functionality is prioritized while allowing for potential enhancements and scalability.

2. Overall Description

The following section provides a comprehensive overview of the Arithmetic Calculator Software, its intended purpose, system interfaces, user interfaces, hardware and software dependencies, and the key functionalities it offers. This section serves as a foundational background for understanding the requirements detailed in Section 3 of the Software Requirements Specification (SRS).

2.1 Product perspective

2.1.1 System Interfaces:

- The calculator provides a user interface for entering mathematical expressions and displaying results. It may interact with the underlying operating system to handle user input and display output on the screen.

2.1.2 User Interfaces:

- The user interface consists of a graphical user interface (GUI) with buttons for digits (0-9) and operators (+, -, *, /), as well as functions like clear, equals, and backspace. Users can input mathematical expressions by clicking on buttons or using a keyboard.

2.1.3 Hardware Interfaces:

- The calculator relies on standard hardware interfaces, including the display screen, keyboard or touchscreen for user input, and system resources for processing calculations.

2.1.4 Software Interfaces:

- The calculator software may utilize libraries or APIs for mathematical operations and user interface components.

2.1.5 Communication Interfaces:

- The calculator does not require external communication interfaces as it operates locally on the user's device.

2.1.6 Memory Constraints:

- The application should be designed to run efficiently within the memory constraints of the target devices, ensuring that it does not consume excessive memory resources.

2.1.7 Operations:

- The calculator performs basic arithmetic operations, including addition, subtraction, multiplication, and division. It also supports clearing the input, calculating results, and correcting input errors.

2.2 Product functions

- The primary function of the calculator is to perform arithmetic calculations accurately and efficiently.
- It should display accurate results after a user inputs mathematical expressions.

2.3 User characteristics

- The calculator is designed for users of all ages who need to perform simple arithmetic calculations.
- It should have an intuitive user interface that is easy to use for individuals with varying levels of technical proficiency.

2.4 Constraints

- The calculator should adhere to mathematical rules and constraints, such as the order of operations (PEMDAS).
- It should be created using the C++ coding language

2.5 Assumptions and dependencies

- The calculator assumes that user inputs are mathematical expressions and may not handle non-standard or complex expressions.

2.6 Requirements subsets

- The specific functional and non-functional requirements for the calculator, including features like decimal support, memory functions (e.g., memory recall), and error handling, are detailed in Section 3 of the software requirements specification (SRS).

3. Specific Requirements

This section of the Software Requirements Specification (SRS) outlines the specific software requirements for the development of an arithmetic calculator in C++. These requirements serve as the foundation for testing the system to ensure it meets its specified criteria.

The overarching goal is to create a user-friendly arithmetic calculator that performs a range of mathematical operations, adheres to the order of operations (PEMDAS), and offers precise handling of various mathematical expressions. This section will define specific functionalities,user interaction, and system behaviors that the calculator must exhibit.

3.1 Functionality

3.1.1 < Addition (Function: add) >

- The system should provide the capability to perform addition operations.
- Users should be able to input two numbers, and the system must accurately add them
- The addition function should handle both positive and negative numbers
- Addition should handle both positive and negative numbers

3.1.2 < Division (Function: divide) >

- The system should support division operators
- Users must be able to input two numbers, and the system should perform accurate division.
- The division operation should handle scenarios such as dividing by zero, providing appropriate error messages when necessary
- Division should handle both positive and negative numbers

3.1.3 < Subtraction (Function: subtract) >

- The system must enable subtraction operations
- Users should be able to input two numbers, and the system should correctly calculate the subtraction result
- Subtraction should handle both positive and negative numbers

3.1.4 < Multiplication (Function: multiply) >

- The system should offer multiplication capabilities

- Users must be able to input two numbers, and the system should accurately perform multiplication
- Multiplication should be able to handle both positive and negative numbers

3.1.5 < PEMDAS (Order of Operations) >

- The system should adhere to the PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) order of operations
- When multiple operations are present in an expression, the system should calculate them following the correct order to ensure accurate results.

3.1.6 < Exponents >

- The system must handle exponential operations
- Users should be able to input a base and an exponent, and the system should calculate the result accurately

3.1.7 < Modulus >

- The system should support modulus operations
- Users must be able to input two numbers, and the system should calculate the modulus (remainder) accurately
- The modulus operation should handle both positive and negative numbers

3.1.8 < Parentheses Handling >

- The system should correctly handle parentheses in mathematical expressions
- Users should be able to group operations within parentheses, and the system must execute them in the correct order to ensure accurate results

3.1.9 < Numerical Constants (Integer to Float Conversion) >

- The system should be able to convert integer values to floating-point numbers when necessary to maintain precision in the calculations
- if a calculation results in a fractional value, the system should ensure that it is represented as a float when output

3.2 Use-Case Specifications

3.2.1 < Use Case 1: Perform Addition >

- Description: This use case involved the addition of two numerical values
- Primary Actor: User
- Trigger: User selects the addition operation
- Scenario:
  - User inputs the first number.
  - User selects the addition operator
  - User inputs the second operation
  - The system calculates and displays the sum
- Alternative Flow: The system handles negative numbers incorrectly
- Postcondition: The result is displayed accurately.

3.2.2 < Use Case 2: Perform division >

- Description: This use case involved the division of two numerical values.
- Primary Actor: User
- Trigger: User selects the division operation
- Scenario:
  - User inputs the dividend (numerator).
  - user selected the division operation.

    - User inputs the divisor (denominator).
    - The system calculates and displays the quotient
- Alternative Flow: The system handles division by zero, displaying an appropriate error message
- Postconditions: The result is displayed accurately.

3.2.3 < Use Case 3: Handle Parentheses >

- Description: This use case involved handling expressions within parentheses
- Primary Actor: User
- Trigger: User includes expression with parentheses
- Scenario:
    - User enters mathematical expressions with parentheses
    - The system calculates expressions within parentheses before executing other operations
- Alternative Flow: They system correctly identifies and evaluates expressions within parentheses, following the order of operations
- Postconditions: Accurate results are displayed for expressions involving parentheses.

3.2.4 < Use Case 4: Exponentiation >

- Description: This use case involves raising a number to an exponent.
- Primary Actor: User
- Trigger: User selects the exponentiation operation.
- Scenario:
    - User inputs the base number.
    - User selects the exponentiation operation.
    - User inputs the exponent.
    - The system calculates and displays the result.
- Alternative Flow: The system handles both positive and negative exponents.
- Postconditions: The result is displayed accurately.

3.2.5 < Use Case 5: Error Handling (Division by Zero) >

- Description: This use case involves handling division by zero.
- Primary Actor: User
- Trigger: User attempts division by zero.
- Scenario:
    - User inputs a divisor (denominator) of zero.
    - The system displays an error message indicating division by zero is not allowed.
- Postconditions: An appropriate error message is displayed.

3.3 Supplementary Requirements

3.3.1 < Non-Functional Requirements: >

- **Error Handling:** The system should have robust error-handling mechanisms to gracefully handle unexpected inputs, ensuring that it provides clear error messages when necessary.
- **Usability:** The user interface should be intuitive, and user interaction should be straightforward. Users should be able to easily understand and navigate the calculator
- **Efficiency:** The calculator should provide prompt responses to user input, ensuring that calculations are performed with minimal delay.
- **User-Friendly Command Line:** The calculator should have a command-line interface that is easy to use and navigate, ensuring positive user experience.

3.3.2 < Constraints: >

- **C++:** The development of the calculator application should be done using the C++ programming language, as specified.
- **Platform:** The calculator application should be designed to run on multiple platforms, including Windows, macOS, and Linux.
- **Performance:** The system should be optimized for performance, with the ability to handle complex calculations efficiently.

4. Classification of Functional Requirements

Functionality	Type
The program should be able to handle an “add” operator.	Essential
The program should be able to handle a “subtract” operator.	Essential
The program should be able to handle a “divide” operator.	Essential
The program should be able to handle a “multiply” operator.	Essential
The program should be able to validate user input.	Essential
The program should be able to backup and recover data.	Desirable
The program should be compatible on several platforms.	Desirable
The program should have aesthetic value.	Optional
The program should be scalable.	Optional

5. Appendices

5.1 Glossary (Refer to Section 1.3)

- Constraints: A part of the non-functional requirements, setting boundaries for how the software will satisfy other requirements.
- Functional requirements: Describe what a system should do, addressing the “what” aspects and characterizing units of functionality that are sometimes referred to as *features*.
- Non-functional requirements: Define constraints on the way software should satisfy its functional requirements, often including performance, security, and portability.
- Use-Case: A sequence of transactions performed by a system which yields an observable result of value for a particular actor

5.2 References (Refer to Section 1.4)

- EECS348: Term Project in C++ Specification from Professor Hossein Saiedian
- Handling divide by zero error from ManasiKirloskar
- Microsoft How to make a calculator on C++ from multiple contributors
- PEMDAS- order of mathematical operations in C++ from Gearbox Labs
- Inspiration for GUI for our calculator from Filipe Salgueiro